CSE5P: Assignment 5 (Due: Wednesday February 20, 11:59PM)

TA: Trung Nguyen, Instructor: Peter Alvaro

Spring 2019

Instructions

Please read the following requirements carefully before coding. You will be penalized **15 points** if you fail to comply with **any** of them:

1. Code must be written in Python3 (NOT Python2)
2. You are asked to download the `hw5.py` file on Canvas and modify it
3. You are only allowed to modify some specific portions of the provided `hw5.py` python files. **DO NOT** modify any existing line of the provided skeleton code. Where you can code and where you cannot is already noted by comments in the python file
4. **DO NOT** put the `hw5.py` in a folder or zip it and **DO NOT** change its file name. Also, `hw5.py` should be the only file you should submit to Canvas, **DO NOT** submit any other files, except `hw5.py`
5. There will be no test script for this assignment because we already provided the inputs and print statements in this file for you to debug your program. Note that we may use a different set of inputs for grading

Question 1. Password Checker [60 pts + 20 pts extra credit]

Brute-force Attack is a technique where hackers try every possible password until they find the correct one. Although moderns computers can try thousands of passwords per second, this technique is only worthwhile when a password is easy to identify, which is why a strong password matters.

As a website administrator, you need to make sure your users have strong passwords. You will need to check the strength of their passwords. A strong password must satisfy **all** of the following rules:

1. The password must have at least 8 characters
2. The password must NOT contain the question mark symbol
3. The password must have at least one numerical, one alphabetical, and one special character (neither numerical, nor alphabetical)
4. The password must not contain the same character more than two times consecutively. For example, "abbbcd1@" is not a strong password
5. The number of distinct characters in the password must not be less than half of the password's length. For example, "a1@a1@1@a" has only three distinct characters, "a", "1" and "@", but the length of the password is 8, thererfore it fails the test

Modify the the function `password_check()` in `hw5.py` so that it takes a password string as its only argument, returns `True` if the password is strong enough, and `False` otherwise. You can assume that the argument for this function is always a string, so no need to check for invalid input type.

20 pts extra credit will be added if your program can pass all tests cases, which are revealed after grades are released.

Example:

```
>>> password_check('a12#8')
False
>>> password_check('a1234567#?')
False
>>> password_check('abcdefgh#')
False
>>> password_check('a12345678')
False
>>> password_check('12345678#')
False
```

```
>>> password_check('abbbcd1@')
False
>>> password_check('a1@a1@1@')
False
>>> password_check('a123456#7')
True
```

Question 2. Bag-Of-Words [40 pts]

Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand and interpret human language. Recent exploration in this field has allowed dramatic growth in development of intelligent personal assistants such as Apple Siri, Amazon Alexa or Google Home.

One of the models used in NLP is Bag-Of-Words, where a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. One of its popular applications is in Spam Email Filtering. Imagine there are two emails represented as two literal bags full of words. One bag is filled with words found in spam messages, and the other with words found in legitimate e-mail. While any given word is likely to be somewhere in both bags, the "spam" bag will contain spam-related words such as "stock", "viagra", and "buy" significantly more frequently, while the "ham" bag will contain more words related to the user's friends or workplace.

In this question, you will implement a simplified version of the bag-of-words model by counting the numbers of occurrence of words in a text file.

You need to modify the functions `word_counter()` and `most_frequent()` in `hw5.py` so that they both take the name of a txt file **located in the same folder as hw5.py** as a single parameter and do the followings:

1. Function `word_counter()` returns an integer indicating the total number of words in the txt file. A word is defined by the blank characters (can be either space or new line) around it.

   For example, consider the following txt files:

   - `input1.txt`

   ```
   The first   second was  alright,

   but the second   second was tough.
   ```

   - `input2.txt`

   ```
   The first   second was  alright ,

   but the second   second was tough.
   ```

   - `input3.txt`

   ```
   In May, May may move to San Mateo.
   ```

   - `input4.txt`

   ```
   Will Smith will smith
   will Smith smith Will?
   ```

   Sample runs:

   ```
   >>> word_counter('input1.txt')
   11
   >>> word_counter('input2.txt')
   12
   >>> word_counter('input3.txt')
   8
   >>> word_counter('input4.txt')
   8
   ```

   Explanation: In the `input2.txt` case the comma symbol is considered a word, so there is one more word than in the `input1.txt` case

2. `Function most_frequent()` creates a dictionary where keys are the words of the document, and the values are the number of occurrence of these words. It returns a tuple containing two values: the first one is the most frequent word in the text file (string type) in lowercase, and the second one is its occurrence (int type).

   Note that the keys should not be case sensitive. For example 'dinosaur' and 'DiNosAuR' should be considered the same word. In case there are multiple keys having the same maximum occurrence then you can return any of them.

   Sample run:

```
>>>most_frequent('input1.txt')
('second', 3)
>>>most_frequent('input2.txt')
('second', 3)
>>> most_frequent('input3.txt')
('may', 2)
>>> most_frequent('input4.txt')
('smith', 4)
```

Explanation: In the `input3.txt` case, 'may' and 'May' are considered the same but 'may,' and 'may' are considered two different words so the count of 'may' is only 2. Similarly, in `input4.txt`, 'will?' and 'will' are considered two different words.

Useful functions/methods

Since it may be intimidating to start solving this assignment from scratch, we are providing you a list of some functions/methods below which can be helpful to solve this assignment:

- isalpha()
- isdigit()
- lower()
- max()
- split()

They are given as a hint so you are not forced to use them and still have the freedom to use any functions/methods you want as long as you don't have to import any external library to call them.